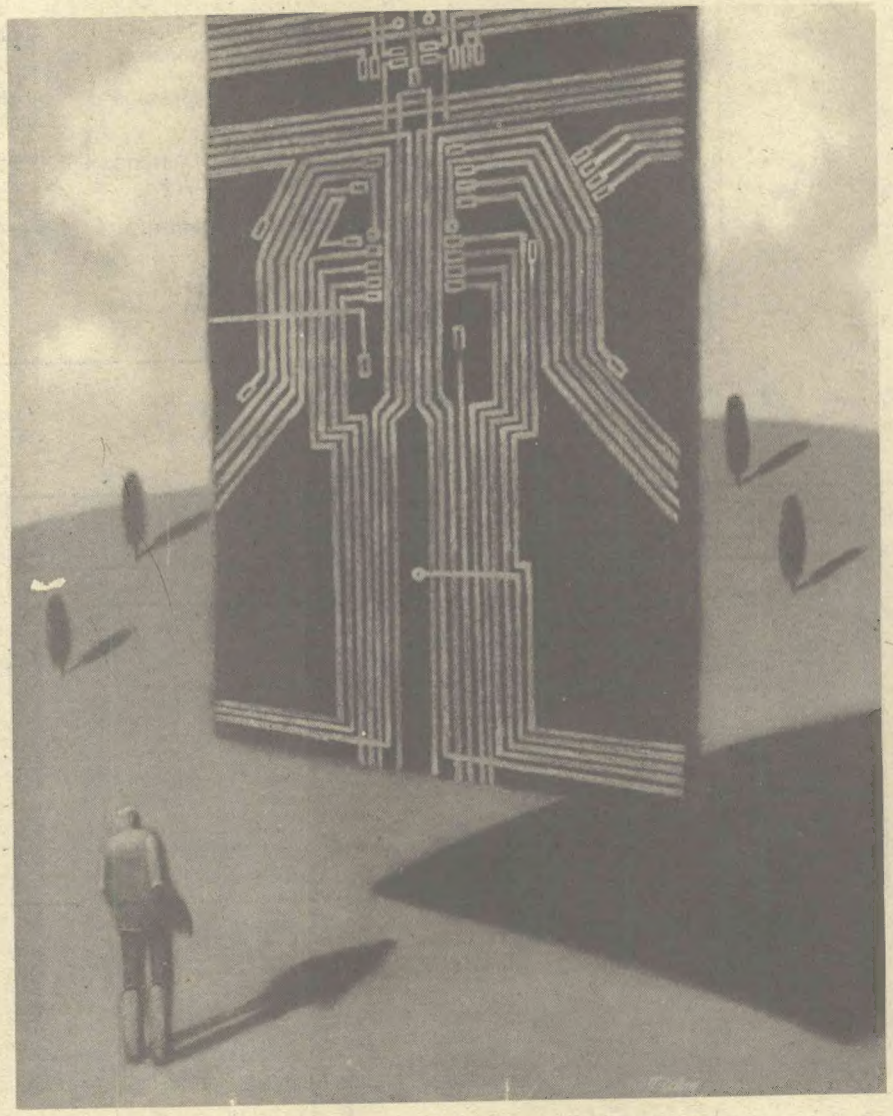
ขจร โรจนเมธินทร์

เขียนโปรแกรมบนวินโดวส์ โดยไม่ใช้ C++



ทางเลือกใหม่ของ การเขียนโปรแกรม ที่ใช้งานบนวินโดวส์ กับการเขียนด้วยแบตช์ไฟล์ พร้อมกับวิธีการใช้งานอย่างง่าย

ผู้อ่านหลายท่านที่ใช้ไมโครซอฟต์-วินโดวส์อยู่ คงอยากเขียนโปรแกรมบนวินโดวส์ขึ้นมาใช้เองบ้าง แต่พอคิดว่าจะต้องใช้ภาษาซีหรือปาสคาล ผู้อ่านที่ไม่ได้ใช้ภาษาซีอยู่ประจำคงรู้สึกว่าเป็นเรื่องยากที่จะต้องเริ่มต้นศึกษาภาษาซีก่อนและยังต้องเรียนรู้วิธีการเขียนโปรแกรมบนวินโดวส์อีก กว่าจะเขียนโปรแกรมขึ้นมาใช้ได้ก็อาจทำให้หมดความพยายามเสียก่อน บทความที่นำมาเสนอในครั้งนี้จะทำให้ผู้อ่าน เขียนโปรแกรมบนวินโดวส์ได้โดยไม่ต้องเรียนภาษาซีและวิธีเขียนโปรแกรมบนวินโดวส์อีกเลย ผู้อ่านสามารถเขียนโปรแกรมที่แสดงผลบนจอภาพได้ดังรูปที่ 1 โดยใช้เวลาไม่เกิน 5 นาที และตัวโปรแกรมจะมีความยาวเพียงหนึ่งบรรทัดเท่านั้น

มารู้จักกับภาษาที่จะใช้ก่อน

การเขียนโปรแกรมบนวินโดวส์ที่ผู้เขียนกล่าวถึงนี้เป็นการเขียนแบตช์ไฟล์ ซึ่งเป็นโปรแกรมชนิดหนึ่งที่ทำงานบนวินโดวส์ได้ แบตช์ไฟล์ที่กล่าวถึงนี้เป็นแบตช์โปรแกรมที่เขียนขึ้นใช้งานบนนอร์ตันเดสค์ท็อป ซึ่งนอร์ตันเดสค์ท็อปได้เตรียมชุดคำสั่งที่จะใช้เขียนโปรแกรมในรูปแบตช์ไฟล์เอาไว้มากมาย สำหรับการเขียนแบตช์โปรแกรมนี้ผู้อ่านสามารถใช้ Batch Builder ที่มีในนอร์ตันเดสค์-ท็อปหรือจะใช้เท็กซ์เอดิเตอร์ตัวอื่นก็ได้ แบตช์ไฟล์ที่สร้างขึ้นนี้ไม่ได้ใช้นามสกุล BAT เหมือนในดอสแต่ใช้นามสกุล WBT โดยแบตช์ไฟล์ที่สร้างขึ้นนี้จะทำงานได้ต้องใช้ Batch Runner ช่วยแปลโปรแกรม

การใช้ Batch Builder

การเรียก Batch Builder มาทำงานนั้นทำได้หลายวิธี ได้แก่ เรียกจากเมนู Tools ของนอร์ตันเดสค์ท็อป, เรียกจากเมนู File โดยเลือก "Run..." หรือเรียกจาก Quick Access ถ้าผู้ใช้ติดตั้งไว้ หลังจากเรียก Batch Builder มาทำงานแล้วจอภาพจะแสดงผลตามรูปที่ 2

หากผู้อ่านต้องการสร้างแบตช์ไฟล์ชุดใหม่ก็ทำได้โดยเลือก "New..." จากเมนู File แล้วใช้ Batch Builder เป็นเอดิเตอร์สำหรับเขียนโปรแกรม (วิธีใช้เหมือนกับ Notepad ของวินโดวส์) หลังจากที่เขียนเสร็จให้ใช้ "Save As..." เก็บไฟล์นั้น ในกรณีที่ต้องการแก้ไขแบตช์ไฟล์ที่มีอยู่แล้วให้เลือก "Open..." และใช้ "Save..." แทน

นอกจากส่วนเอดิเตอร์นี้แล้ว Batch Builder ยังเตรียมเครื่องมือสำหรับช่วยเขียนแบตช์โปรแกรมให้สะดวกขึ้นคือ รีเฟอเรนซ์ไดอะล็อกบ็อกซ์ หากผู้อ่านเลือกเมนู Reference! จอภาพจะแสดงผลตามรูปที่ 3 ซึ่งจะเห็นว่า Reference จะประกอบด้วยกรอบ Commands ที่แสดงคำสั่งที่มีทั้งหมดให้ผู้ใช้เลือกดูโดยจะมีคำอธิบายของแต่ละคำสั่งแสดงอยู่ในกรอบ Description หลังจากที่ผู้ใช้เลือกคำสั่งจากกรอบ Commands แล้ว ผู้อ่านสามารถแทรกคำสั่งที่เลือกในเอดิเตอร์ได้ทันทีโดยคลิ๊กเม้าส์ที่ "Add" หรือคลิ๊กเม้าส์สองครั้งที่คำสั่งที่ต้องการในกรอบ Commands เลย แต่ถ้าต้องการให้คำสั่งที่เลือกนั้นแทนบรรทัดที่อยู่ในเอดิเตอร์เลยก็ต้องทำบรรทัดนั้นให้อยู่ในแถบสีดำก่อนแล้วคลิ๊กเม้าส์ที่ "Add" หรือคลิ๊กเม้าส์สองครั้งที่คำสั่งเช่นกัน

การใช้ Batch Runner

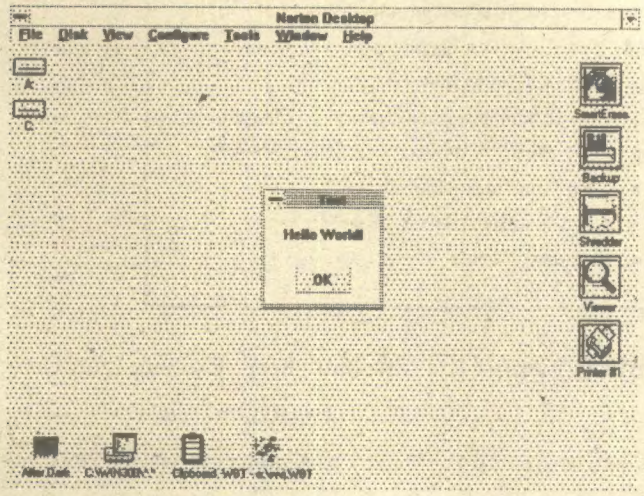
แบตช์ไฟล์ที่สร้างขึ้นจาก Batch Builder นี้ จะต้องใช้ Batch Runner มาช่วยให้แบตช์โปรแกรมทำงานได้ โดยขณะที่แบตช์ไฟล์กำลังทำงานจะมีไอคอนของ Batch Runner ปรากฏอยู่ การเรียกแบตช์ไฟล์มาทำงานทำได้ 4 วิธี คือ

- ใช้ "Run..." จากเมนู File โดยใส่ชื่อแบตช์ไฟล์ที่ต้องการ ซึ่งถ้าผู้ใช้ไม่ได้กำหนดไดเรคทอรีที่เก็บแบตช์ไฟล์ด้วยคำสั่ง Path ของดอส ก็จะต้องกำหนดไว้ด้วย
- เรียกแบตช์ไฟล์จากไดรฟ์วินโดวส์ โดยคลิ๊กเม้าส์สองครั้งที่ชื่อแบตช์ไฟล์ที่ต้องการจะให้ทำงาน
- ผู้ใช้อาจสร้างกลุ่มวินโดวส์ที่เก็บแบตช์ไฟล์ต่าง ๆ ไว้ แล้วเรียกแบตช์ไฟล์มาทำงานโดยคลิ๊กเม้าส์สองครั้งที่ชื่อแบตช์ไฟล์
- ใช้ Launch Manager โดยเพิ่มชื่อแบตช์ไฟล์เข้าไปใน Launch List นอกจากนี้ผู้ใช้อาจกำหนด Shortcut Key ได้ด้วย

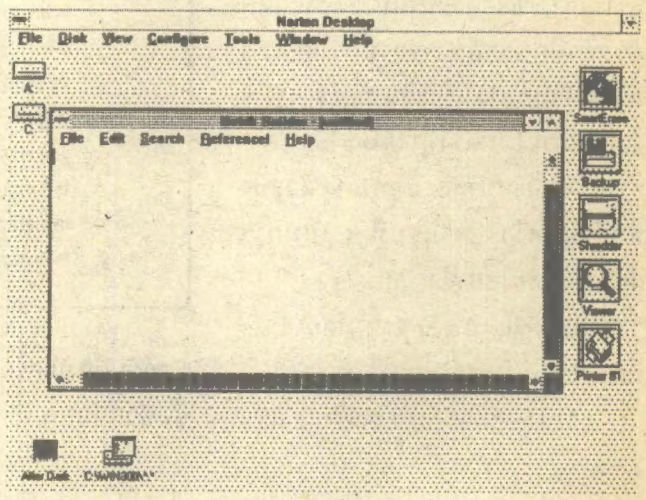
แบตช์ไฟล์จะหยุดการทำงานเมื่อเสร็จสิ้นการทำงาน, เมื่อมีข้อผิดพลาดของโปรแกรม หรือเมื่อเลือก "Close" จากคอนโทรลเมนู

รายละเอียดของแบตช์โปรแกรม

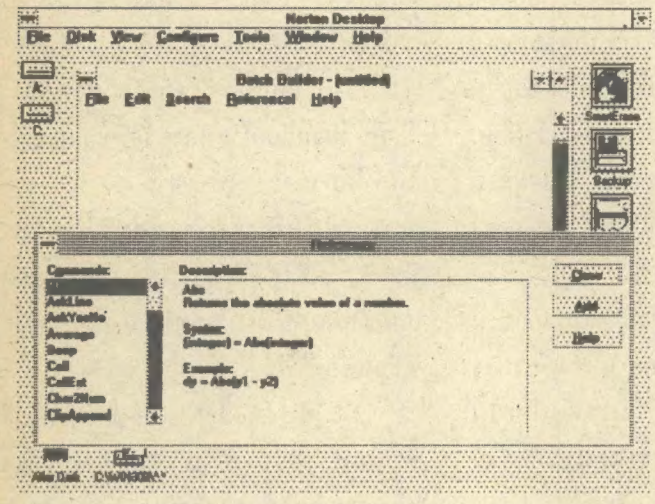
โปรแกรมที่ถูกเขียนขึ้นนั้นจะประกอบด้วยประโยคต่าง ๆ ซึ่งจะแยกออกเป็นส่วน ๆ ได้ดังนี้



รูปที่ 1 แสดงจอภาพ เมื่อใช้คำสั่ง Message ("Test", "Hello World!")



รูปที่ 2 แสดงจอภาพเมื่อเรียก Batch Builder จากเมนู Tools



รูปที่ 3 แสดงจอภาพเมื่อเลือกเมนู Reference!

รูปที่ 4 แสดงการใช้ตัวแปรในสตริงของคำสั่ง Message (" ", "Tank you, % user %) โดย user = "Jame"

1. ค่าคงที่ (Constant) ซึ่งใช้ได้ในแบตช์ไฟล์นี้จะมีอยู่สามประเภทคือ

– จำนวนเต็ม (Integer) ซึ่งประกอบด้วยเลข 0 ถึง 9 เรียงกัน โดยจะมีค่าอยู่ในช่วง $-2^{31}+1$ ถึง $2^{31}-1$

– สตริง (String) ประกอบด้วยคาแรกเตอร์ซึ่งอยู่ในเครื่องหมาย ","

– ค่าคงที่ซึ่งกำหนดค่าไว้แล้ว ได้แก่ @FALSE, @NO, @STACK, @TILE, @TRUE, @YES

2. ตัวแปร (Variables) จะใช้เก็บค่าคงที่ทั้งสามชนิด ขณะที่โปรแกรมทำงานการตั้งชื่อตัวแปรนั้นจะต้องขึ้นต้นด้วยตัวอักษรแล้วอาจตามด้วยตัวเลขหรือตัวอักษรยาวได้ไม่เกิน 30 ตัว การใช้ตัวแปรภายในสตริงทำได้โดยใช้เครื่องหมาย % นำหน้าและปิดท้ายชื่อตัวแปรที่อยู่ในสตริง จากรูปที่ 4 เป็นการแสดงผลเมื่อใช้คำสั่ง Message (" ", "Thank you, %user%") โดย User มีค่า "Jame"

3. Operators แบ่งออกเป็น 2 กลุ่ม ดังรูป A

- Unary operators ใช้กระทำกับตัวแปรหนึ่งตัวได้แก่
  - a. เครื่องหมาย – ใช้เปลี่ยนเป็นจำนวนเต็มลบแบบ Two's Com
  - b. เครื่องหมาย + ใช้เปลี่ยนเป็นจำนวนเต็มบวก
  - c. เครื่องหมาย ~ ใช้เปลี่ยนค่าของบิตจาก 0 เป็น 1 หรือกลับกัน
  - d. เครื่องหมาย ! ใช้เปลี่ยนค่าตรรกจริงเป็นเท็จหรือกลับกัน
- Binary operators ใช้กระทำกับตัวแปรสองตัวได้แก่
  - a. เครื่องหมาย / ใช้หาผลลัพธ์ของการหารแบบจำนวนเต็ม
  - b. เครื่องหมาย mod ใช้หาเศษของการหารแบบจำนวนเต็ม
  - c. เครื่องหมาย + ใช้หาผลบวก
  - d. เครื่องหมาย – ใช้หาผลลบ
  - e. เครื่องหมาย << ใช้เลื่อนบิตไปทางซ้าย
  - f. เครื่องหมาย >> ใช้เลื่อนบิตไปทางขวา
  - g. เครื่องหมาย & ใช้กระทำ AND แบบคณิตศาสตร์
  - h. เครื่องหมาย | ใช้กระทำ OR แบบคณิตศาสตร์
  - i. เครื่องหมาย ^ ใช้กระทำ XOR แบบคณิตศาสตร์
  - j. เครื่องหมาย && ใช้กระทำ AND แบบตรรกศาสตร์
  - k. เครื่องหมาย || ใช้กระทำ OR แบบตรรกศาสตร์
  - l. เครื่องหมาย > , >= ใช้เปรียบเทียบค่ามากกว่าและมากกว่า หรือเท่ากับ
  - m. เครื่องหมาย < , <= ใช้เปรียบเทียบค่าน้อยกว่าและน้อยกว่า หรือเท่ากับ
  - n. เครื่องหมาย == ใช้เปรียบเทียบค่ากรณีที่เท่ากับ
  - o. เครื่องหมาย != , < > ใช้เปรียบเทียบค่าธรณีที่ไม่เท่ากับ
  - p. เครื่องหมาย = ใช้กำหนดค่าให้ตัวแปร

รูป A

4. Functions ฟังก์ชันที่ใช้ในแบตช์โปรแกรมมีให้เลือกมากมาย การใช้ฟังก์ชันต้องมีเครื่องหมาย ( ) ตามหลังชื่อฟังก์ชันเสมอ

5. Commands เป็นกลุ่มคำสั่งซึ่งใช้ในแบตช์โปรแกรมโดยไม่ต้องมีวงเล็บตามหลังและไม่สามารถกำหนดค่าได้ ได้แก่ Beep, Execute, Exit, Goto, If...Then, Return, Yield

6. Comments จะใช้เครื่องหมาย ; นำหน้าประโยค

รูป A1

กลุ่มคำสั่งที่มีในแบตช์ไฟล์

กลุ่มที่ 1 ใช้สำหรับเรียกโปรแกรมอื่นมาทำงาน

- Run(program-name, parameters)
  เรียกโปรแกรมมาทำงานในขนาดวินโดวส์ปกติ
- RunHide(program-name,parameters)
  เรียกโปรแกรมโดยซ่อนวินโดวส์ไว้
- RunIcon (program-name,parameters)
  เรียกโปรแกรมมาทำงานเป็นไอคอน
- RunZoom(program-name,parameters)
  เรียกโปรแกรมมาทำงานในวินโดวส์ขนาดใหญ่สุด

กลุ่มที่ 2 ใช้ถามการตอบรับจากผู้ใช้

- AskLine(title,prompt,default)
  ใช้สอบถามข้อความจากผู้ใช้
- AskYesNo(title,question)
  ให้ผู้ใช้เลือกคำตอบระหว่าง Yes, No หรือ Cancel
- ItemSelect(title,list,delimiter)
  ให้ผู้ใช้เลือกไอเทมจากลิสต์บอก
- TextBox(title,filename)
  แสดงลิสต์บอกและให้ผู้ใช้เลือก

กลุ่มที่ 3 ใช้ให้ข่าวสารกับผู้ใช้

- Beep
  ส่งเสียงเตือนผู้ใช้หนึ่งครั้ง
- DialogBox(title,WBD-file)
  เปิดไดอะล็อกบนวินโดวส์โดยใช้โครงสร้างจากไฟล์นามสกุล WBD
- Display (seconds,title,text)
  แสดงข้อความในช่วงเวลาที่กำหนด
- ItemCount(list,delimiter)
  หาจำนวนไอเท็มในลิสต์
- ItemExtract(item-position,list,delimiter)
  เลือกไอเท็มที่ต้องการจากลิสต์
- Message(title,message)
  แสดงข้อความจนกระทั่งผู้ใช้คลิ๊กเม้าส์ที่ OK
- Pause(title,message)
  แสดงข้อความจนกระทั่งผู้ใช้คลิ๊กเม้าส์ที่ OK หรือ Cancel

กลุ่มที่ 4 ใช้จัดการกับไฟล์

- FileClose(filehandle)
  ปิดไฟล์ที่เปิดโดยใช้ค่า filehandle ที่ได้จาก FileOpen
- FileCopy(from-list,destination,warning)
  ก๊อบปี้ไฟล์โดยมีหรือไม่มีข้อความเตือนถ้าจะต้องมีการเขียนทับ
- FileDelete(file-list)
  ลบไฟล์โดย file-list จะใช้เป็น Wildcards ก็ได้
- FileExist(filename)
  ใช้ทดสอบว่ามีไฟล์ที่ต้องการอยู่หรือไม่
- FileExtension(filename)
  ใช้หาชนิดนามสกุลของไฟล์
- FileItemize(file-list)
  ใช้สร้างลิสต์ของชื่อไฟล์ซึ่งกำหนดกลุ่มของไฟล์ด้วย file-list
- FileLocate(filename)
  ใช้ค้นหาเส้นทางที่อยู่ของไฟล์
- FileMove(from-list,destination,warning)
  ย้ายไฟล์โดยมีหรือไม่มีข้อความเตือนถ้าจะต้องมีการเขียนทับ
- FileOpen(filename,read-or-write)
  เปิดไฟล์เพื่ออ่านหรือเขียน โดยจะคืนค่า filehandle ให้
- FilePath(filename)
  ตัดเอาเฉพาะชื่อซับไดเรคทอรีจากตัวแปร filename
- FileRead(filehandle)
  อ่านข้อมูลจากไฟล์ที่เปิดไว้ โดยใช้ filehandle
- FileRename(from-list,destination)
- FileRoot(filename)
  ตัดเอาเฉพาะชื่อไฟล์(ไม่รวมนามสกุล) จากตัวแปร filename
- FileSize(file-list)
- FileWrite(filehandle,output-data)
- IniRead(section,keyname,default)
  อ่านข้อความจากไฟล์ WIN.INI ถ้าหาไม่พบจะคืนค่า default
- IniReadPvt(section,keyname,default,filename)
  อ่านข้อความจากไฟล์นามสกุล INI ถ้าหาไม่พบจะคืนค่า default
- IniWrite(section,keyname,string)
  เขียนข้อความในไฟล์ WIN.INI
- IniWritePvt(section,keyname,string,filename)
  เขียนข้อความในไฟล์นามสกุล INI

กลุ่มที่ 5 ใช้จัดการกับไดเรคทอรี

- DirChange([d:]path)
- DirGet( )
- DirHome( )
  ใช้หาซับไดเรคทอรีที่เก็บ Batch Runner
- DirItemize(dir-list)
  ใช้สร้างลิสต์ของชื่อซับไดเรคทอรีซึ่งจะกำหนดด้วย file-list
- DirMake([d:]path)
- DirRemove(dir-list)
- DirRename([d:]oldpath,[d:]newpath)

กลุ่มที่ 6 ให้จัดการกับไดรฟ์

- DiskFree(drive-list)
- LogDisk(drive)

กลุ่มที่ 7 ใช้จัดการกับวินโดวส์

- WinActivate(partial-window-name)
  ทำให้วินโดวส์ที่กำหนดหรือไอคอนเป็นแอคติฟวินโดวส์
- WinArrange(style)
  จัดวิธีการเรียงวินโดวส์ทั้งหมดที่ทำงานอยู่บนเดสค์ท็อป
- WinClose(partial-window-name)
  ปิดวินโดวส์ที่กำหนดหรือไอคอน
- WinConfig( )
  คืนค่าผลรวมของบิตของวินโดวส์คอนฟิกูเลชัน (Window configuration)
- WinExist(partial-window-name)
  ทดสอบว่าวินโดวส์ที่กำหนดหรือไอคอนมีอยู่หรือไม่
- WinGetActive( )
  ใช้ถามไตเติ้ลของแอคติฟวินโดวส์หรือไอคอน
- WinHide(partial-window-name)
  ใช้ซ่อนวินโดวส์หรือไอคอน
- WinIconize(partial-window-name)
  ทำให้วินโดวส์ที่กำหนดเป็นไอคอน
- WinItemize( )

รูป A1 (ต่อ)

ใช้สร้างลิสต์ของชื่อวินโดวส์ทั้งหมดที่เปิดอยู่

- WinPlace(x-ul,y-ul,x-br,y-br,partial-window-name)
  ใช้เปลี่ยนตำแหน่งและขนาดของวินโดวส์
- WinPosition(partial-window-name)
  ใช้หาตำแหน่งของวินโดวส์ที่อยู่บนเดสค์ท็อป
- WinShow(partial-window-name)
  นำวินโดวส์หรือไอคอนขึ้นมาแสดงตามขนาดปกติ
- WinTitle(partial-window-name,new-window-name)
  เปลี่ยนไตเติ้ลของวินโดวส์หรือไอคอน
- WinWaitClose(partial-window-name)
  ใช้รอให้ผู้ใช้ปิดวินโดวส์หรือไอคอนก่อน แบตช์ไฟล์จึงจะจบการทำงานได้
- WinZoom(partial-window-name)
  ขยายขนาดวินโดวส์หรือไอคอนให้ใหญ่ที่สุด

กลุ่มที่ 8 ใช้จัดการกับสตริง

- Char2Num(string)
- IsNumber(string)
- Num2Char(number)
- ParseData(string)
  ใช้แบ่งสตริงออกเป็นส่วนๆ เก็บในตัวแปร param1, param2,... ด้วยช่องว่าง และ param0 จะเก็บจำนวนที่แบ่งได้
- StrCat(string1,string2[,stringN]...)
- StrCmp(string1,string2)
- StrFill(substring,string-length)
  สร้างสตริงให้ยาวตามที่กำหนดโดยใช้ substring
- StrFix(base-string,pad-string,length)
  ใช้ต่อสตริงด้วย pad-string หรือตัดสตริงให้ยาวตามที่กำหนด
- StrICmp(string1,string2)
  เปรียบเทียบสตริงโดยไม่สนใจ Case
- StrIndex(string,substring,start-position,search-direction)
- StrLen(string)
- StrLower(string)
- StrReplace(string,old,new)
- StrScan(string,delim-list,start-position,direction)
  ค้นหาตำแหน่งของ delimiter ที่กำหนดจากสตริง
- StrSub(string,start-position,length)
  ตัดเอาส่วนที่ต้องการจากสตริง
- StrTrim(string)
  ตัดช่องว่างที่นำหน้าและตามหลังสตริงออก
- StrUpper(string)

กลุ่มที่ 9 ใช้สำหรับหาผลลัพธ์ทางคณิตศาสตร์

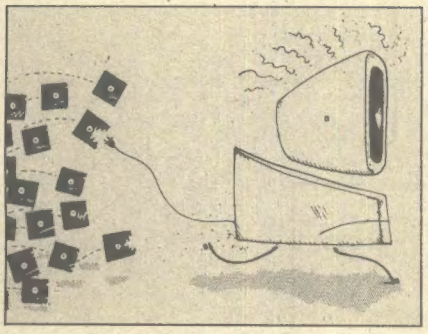
- Abs(integer)
- Average(integer[,integer]...)
- Max(integer[,integer]...)
- Min(integer[,integer]...)
- Random(max)

กลุ่มที่ 10 ใช้จัดการกับคลิปบอร์ดวินโดวส์

- ClipAppend(string)
  นำสตริงไปเก็บต่อท้ายในคลิปบอร์ด
- ClipGet( )
  คืนสตริงที่เก็บอยู่ทั้งหมดในคลิปบอร์ด
- ClipPut(string)
  นำสตริงไปเก็บในคลิปบอร์ดแทนสตริงที่มีอยู่เดิมทั้งหมด

กลุ่มที่ 11 ใช้ควบคุมระบบ

- Call(filename.wbt,parameters)
  เรียกแบตช์ไฟล์อื่นมาทำงานในฐานะซับรูทีน
- CallExt(filename,wbt,parameters)
  เรียกแบตช์ไฟล์อื่นมาทำงานในฐานะโปรแกรมย่อย
- DateTime( )
- Debug(mode)
  เปิดหรือปิดโหมดดีบัก
- Delay(seconds)
- DOSVersion(level)
  ถามรุ่นของดอสโดยกำหนดค่า level เป็น major, minor
- Drop(var[,var]...)
  ยกเลิกการใช้ตัวแปรจากหน่วยความจำ
- EndSession( )
  จบการทำงานของนอร์ตันเดสค์ท็อปและวินโดวส์ 3.0
- Environment(env-variable)
  คืนค่าของตัวแปรเอนไวโรนเมนต์ของดอสที่กำหนดด้วย SET
- ErrorMode(mode)
- Exclusive(mode)
  กำหนดเวลาที่แบตช์ไฟล์ใช้ทำงาน ถ้ากำหนด mode เป็น @ON จะทำให้แบตช์ไฟล์ทำงานเร็วขึ้น
- Execute[[statement]
  ใช้เอ็กซิคิวต์คำสั่งในโหมดโปรเทค ซึ่งถ้ามีข้อผิดพลาดใดเกิดขึ้นก็จะมีส่วนจัดการกับข้อผิดพลาดนั้นให้
- Exit
- Goto[[label]]
- If [[condition]] Then [[statement]]
- IgnoreInput(mode)
  ใช้เปิดหรือปิดฮาร์ดแวร์อินพุตของวินโดวส์
- IsDefined(variable)
  ใช้ตรวจสอบว่าตัวแปรดังกล่าวถูกกำหนดใช้หรือยัง
- IsKeyDown(key-codes)
  ใช้ตรวจสอบว่ามีการกดคีย์ Shift, Ctrl หรือคลิ๊กเม้าส์หรือไม่
- LastError( )
- Return
- SendKey(character-codes)
  ใช้แทนการกดคีย์บอร์ดตามคีย์ที่กำหนดส่งให้แอคตีฟวินโดวส์
- SKDebug(mode)
  ควบคุมการทำงานของ SendKey ในโหมดดีบัก
- Version( )
  ถามเลขรุ่นของ Batch Runner
- WallPaper(bmp-name,tile)
  เปลี่ยนวอลเปเปอร์ของเดสค์ท็อป
- WinVersion(level)
- Yield
  แบ่งเวลาให้วินโดวส์อื่นทำงาน การใช้ Yield หนึ่งคำสั่งจะแสดงข้อความได้ 20 ข้อความ



```
textfile  = FileItemize ("*.doc *.txt")
yourfile = ItemSelect ("Select the file to edit", textfile
           ,"  ")
Run ("notepad.exe", yourfile)
```

รูป B

```
textfile = FileItemize ("*.doc *.txt")
: retry
yourfile = ItemSelect ("Select the file to edit",textfile, "  ")
If yourfile == "  " Then Goto retry
Run ("notepad.exe",yourfile)
```

รูป C

ลองเขียนแบตช์โปรแกรม

ต่อจากนี้ไปผู้เขียนจะแสดงตัวอย่างการเขียนแบตช์ไฟล์ที่ทำงานบนวินโดวส์เป็นขั้น ๆ สองตัวอย่าง ตัวอย่างแรกจะเป็นแบตช์ไฟล์ที่ใช้สำหรับเรียกใช้โปรแกรม Notepad โดยแบตช์ไฟล์จะช่วยค้นหาไฟล์นามสกุล DOC, TXT ให้ผู้เลือกขึ้นมาแก้ไข ส่วนตัวอย่างที่สองเป็นแบตช์ไฟล์ที่ให้ผู้ใช้เลือกเอดิเตอร์สำหรับการแก้ไขไฟล์ โดยจะใช้ไฟล์โครงสร้างที่มีนามสกุล WBD

เริ่มตัวอย่างที่ 1 ให้ผู้อ่านเรียก Batch Builder มาแล้วสร้างแบตช์ไฟล์ที่มีคำสั่งดังรูป B

หลังจากที่สร้างเสร็จเมื่อลองรันแบตช์ไฟล์จะได้ผลตามรูปที่ 5 ตัวอย่างนี้ใช้คำสั่ง FileItemize สร้างลิสต์ของชื่อไฟล์ที่นามสกุล DOC และ TXT โดยเก็บในตัวแปร textfile จากนั้นจึงใช้ Item Select แสดงชื่อของไฟล์ทั้งหมดในลิสต์และให้ผู้ใช้เลือกไฟล์ที่ต้องการมาเก็บใน yourfile เพื่อส่งให้ Notepad คราวนี้ถ้าผู้ใช้เกิดคลิ๊กเม้าส์ที่ OK โดยไม่เลือกไฟล์ใดเลย ItemSelect จะคืนค่า null string ดังนั้นจึงควรตรวจสอบเงื่อนไขก่อนดังรูป C

```
DirChange ("\")
subdir = DirItemize ("*")
targdir = ItemSelect ("Select directory",subdir,"  ")
DirChange (targdir)
textfile = FileItemize ("*.doc *.txt")
: retry
yourfile = ItemSelect ("Select the file to edit",textfile,"  ")
If yourfile == "  " Then Goto retry
Run ("notepad.exe",yourfile)
```

รูป D

```
DirChange ("\")
:getdir
subdir = DirItemize ("*")
If subdir == "  " Then Goto getfile
targdir = ItemSelect ("Select directory (OK = current)",
          subdir,"  ")
If targdir == "  " Then Goto getfile
DirChange (targdir)
Goto getdir
:getfile
textfile = FileItemize ("*.doc *.txt")
:retry
yourfile = ItemSelect ("Select the file to edit",textfile,"  "
If yourfile == "  " The Goto retry
Run ("notepad.exe",yourfile)
```

รูป E

จากนั้นจะเพิ่มคำสั่ง DirItemize เพื่อเปลี่ยนเส้นทางในการเลือกไฟล์ โดยคำสั่งนี้จะลิสต์ที่เก็บชื่อซับไดเรคทอรีทั้งหมดที่มีอยู่ในไดเรคทอรีปัจจุบัน และใช้คำสั่ง DirChange เพื่อเปลี่ยนไปยังซับไดเรคทอรีที่ต้องการ ดังรูป D

จากส่วนโปรแกรมที่เพิ่มขึ้นมาจะเห็นว่าโปรแกรมจะค้นหาในซับไดเรคทอรีเพียงระดับเดียว ดังนั้นหากต้องการค้นหาในระดับที่ลึกลงไปก็จะต้องสร้างลูปเพิ่มต่อดังรูป E

```
[zoom+1Start editor zoomed]
[backup+Save backup of file]
[editor^1Notepad]        [editor^2WinEdit]
[editor^3Write]          [editor^4WinWord]
```

รูป F

```
file = ItemSelect ("  ",FileItemize ("*.*"),"  ")
zoom = 1
DialogBox ("Edit a file", "edit.wbd")
If backup == 0 Then Goto nobackup
filebackupname = StrCat (FileRoot (file), ".", "bak")
FileCopy (file, filebackupname, @TRUE)
: nobackup
If zoom == 0 Then runcmd = "Run"
If zoom == 1 Then runcmd = "RunZoom"
If editor == 1 Then %runcmd% ("notepad.exe", file)
If editor == 2 Then %runcmd% ("winedit.exe", file)
If editor == 3 Then %runcmd% ("write.exe", file)
If editor == 4 Then %runcmd% ("winword.exe", file)
```

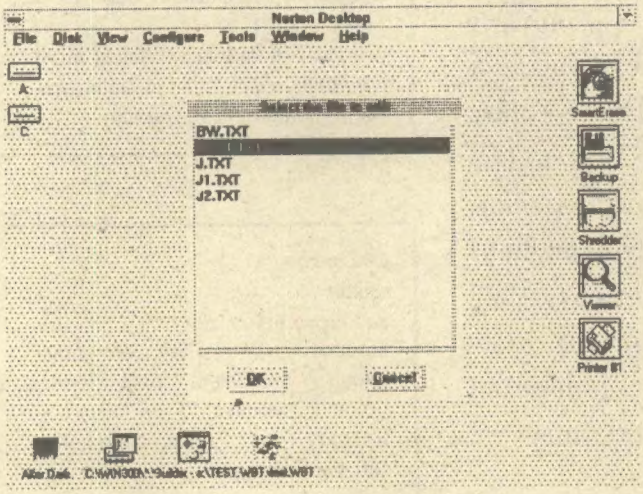
รูป G

เมื่อรันแบตช์ไฟล์ที่แก้ไขข้างต้นแล้ว จอภาพจะแสดงผลตามรูปที่ 6 ต่อจากนี้จะเป็นตัวอย่างที่ 2 ซึ่งจะใช้ไฟล์โครงสร้างที่จะใช้กับคำสั่ง DialogBox โดยไฟล์นี้ใช้นามสกุล WBD ผู้อ่านจะต้องสร้างไฟล์โครงสร้างดังกล่าวเป็นเท็กซ์ไฟล์ดังรูป F โดยให้ชื่อไฟล์เป็น EDIT.WBD

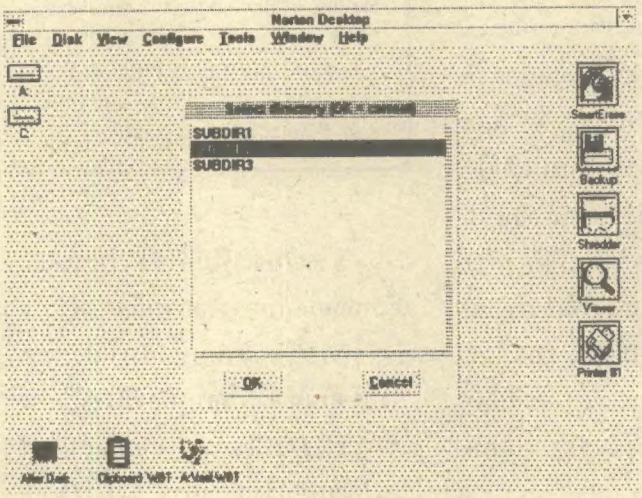
ทุกบรรทัดในไฟล์โครงสร้างนี้จะต้องอยู่ในวงเล็บ [ ] สำหรับ zoom, backup เป็นตัวแปรที่จะเก็บค่าถ้าผู้ใช้เลือก เครื่องหมาย + แสดงว่าบรรทัดนั้นเป็นเช็คบ็อกซ์ (Check Box) สำหรับเลข 1 จะเป็นค่าซึ่งจะกำหนดให้กับตัวแปรถ้าตัวแปรถูกเลือกเครื่องหมาย ^ ใช้แทน Button และเลข 1, 2, 3, 4 จะใช้กำหนดค่าถ้าผู้ใช้เลือกต่อไปขอให้ผู้อ่านสร้างแบตช์ไฟล์ดังรูป G

เมื่อสร้างเสร็จแล้วทดลองรันจะได้ผลตามรูปที่ 7 ต่อไปจะเพิ่มการใช้ตัวแปรอีกสามชนิดคือตัวแปรที่ตามด้วยเครื่องหมาย #แทน file mask ซึ่งรอให้ผู้ใช้ใส่ข้อมูลซึ่งในที่นี้คือ ชื่อไฟล์ที่ต้องการเครื่องหมาย $ ใช้สำหรับแสดงค่าตัวแปรที่อยู่ภายในไฟล์โครงสร้าง, เครื่องหมาย \ ใช้แสดงชื่อที่เก็บในลิสต์ภายในลิสต์บ็อก ต่อไปให้ผู้อ่านเพิ่มจำนวนโปรแกรมในรูป H ในไฟล์ EDIT.WBD

ถ้าทดลองรันแบตช์ไฟล์อีกครั้งจะได้ผลลัพธ์ตามรูปที่ 8



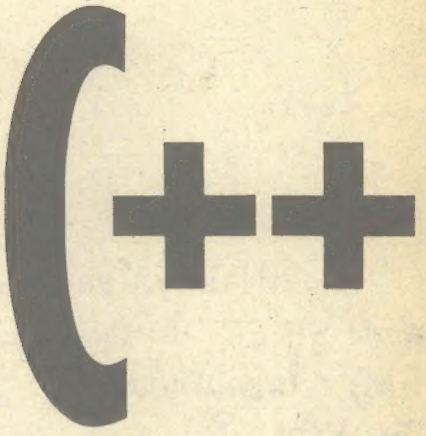
รูปที่ 5
แสดงจอภาพเมื่อเรียกแบตช์ไฟล์ตัวอย่างที่ 1 มาทำงาน

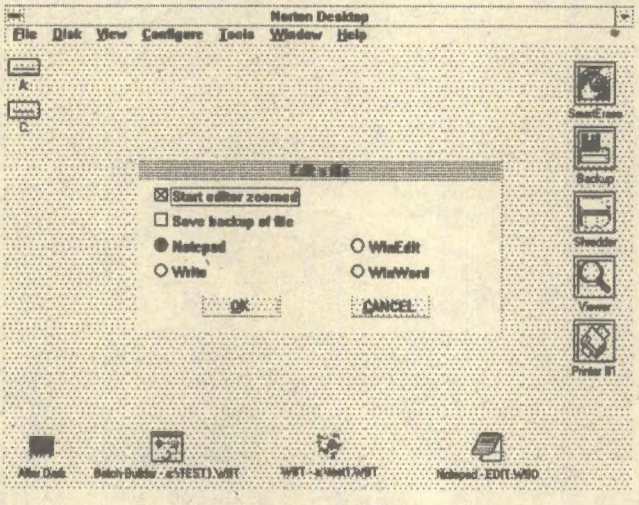


รูปที่ 6
แสดงจอภาพหลังจากปรับปรุงแบตช์ไฟล์ในตัวอย่างที่ 1 แล้ว

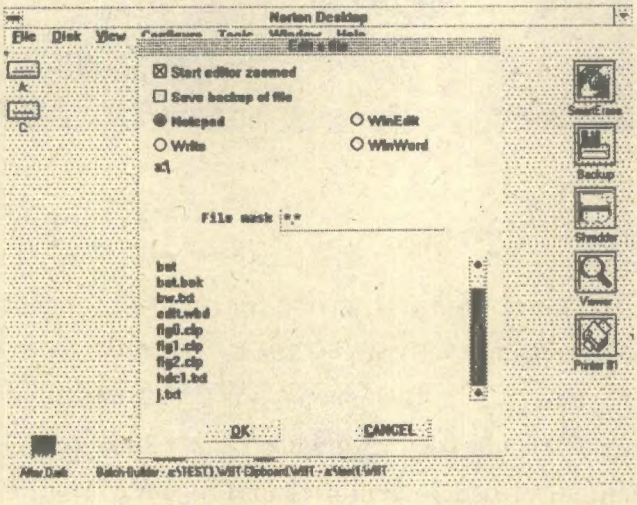
```
[file$]
      File mask [file#                                    ]
[file\                                                    ]
[file\                                                    ]
[file\                                                    ]
[file\                                                    ]
[file\                                                    ]
[file\                                                    ]
```

รูป H





รูปที่ 7
แสดงการใช้ไฟล์โครงสร้างแบบ Check box และ Button โดยใช้คำสั่ง Dialog Box



รูปที่ 8
แสดงการใช้ไฟล์โครงสร้างโดยใช้เครื่องหมาย $, #, \

สรุปตอนท้าย

จากตัวอย่างแบตช์ไฟล์ที่ผู้เขียนนำมาแสดงนี้ ผู้อ่านคงจะเห็นแล้วว่าการเขียนโปรแกรมให้ทำงานบนวินโดวส์นั้นทำได้ง่ายมากเมื่อใช้ Batch Runner ของนอร์ตันเดสค์ท็อป ผู้อ่านไม่ต้องเสียเวลาศึกษาภาษาซีที่ค่อนข้างมากสำหรับการเริ่มต้น แต่อย่างไรก็ตาม ผู้เขียนจะไม่เปรียบเทียบระหว่างโปรแกรมที่เขียนด้วยภาษาซีและซอฟต์แวร์ดีเวลลอปเม้นต์คิตกับแบตช์ไฟล์ เนื่องจากการเขียนแบตช์ไฟล์คงไม่สามารถใช้สร้างโปรแกรมที่มีความซับซ้อนมาก ๆ ได้เหมือนที่เขียนด้วยภาษาซี แต่สำหรับแบตช์ไฟล์ของแบตช์ไฟล์ของนอร์ตันเดสค์ท็อปนี้ ถ้าจะมองในแง่ความสะดวกและง่ายสำหรับการเขียนแล้วผู้เขียนคิดว่าเป็นสิ่งที่เหมาะสมไม่ว่าผู้อ่านจะเป็นโปรแกรมเมอร์มือใหม่หรือผู้ชำนาญในการเขียนโปรแกรม